REMARKS

This paper responds to the Office Action of June 19, 2003. The shortened statutory period runs through September 19, 2003. A one-month Petition for Extension of Time filed herewith renders this response timely.

Claims 1-38 are now pending, a total of 38 claims. Claims 1 and 28 are independent. The Office Action indicates that claim 23 recites allowable subject matter.

Applicant respectfully requests reconsideration of the application.

Formal drawings, a Preliminary Amendment, and an Information Disclosure Statement were submitted November 1, 2000. Receipt and consideration of these papers are not mentioned in the Office Action, and have not been acknowledged by the Examiner.

- The references submitted with the November 2000 IDS are submitted again with the IDS and Form 1449 submitted herewith.
- A replacement copy of the November 2000 Preliminary Amendment is enclosed herewith. I certify that this Preliminary Amendment was deposited with the United States Postal Service on November 1, 2000 as First Class Mail in an envelope with sufficient postage addressed to The Commissioner for Patents, Washington D.C. 20231, and is entitled to entry pursuant to 37 C.F.R. § 1.8. Also enclosed is a copy of the postcard for Preliminary Amendment.
- The Examiner is requested to confirm that formal drawings were received in or around November 2000.

I. § 112 and drawing issues

The Office Action questions the use of the term "effect" in claim 25. Webster's Ninth New Collegiate Dictionary gives the following definitions for "effect" that are applicable to claim 25:

> **effect** *vt* [verb, transitive] 1: to cause to come into being 2a: to bring about often by surmounting obstacles : ACCOMPLISH b: to put into effect

Applicant submits that this usage is grammatically proper.

The existing drawings show every feature of the invention specified in the claims, as follows:

- Examples of a "number of bits used to record the event code" is shown at reference 436 in Fig. 4c, and reference 446 of Fig. 4d. In both cases, the event code is recorded as four bits. A "number of distinguished event classes" is shown as thirty-two (or thirty-one, depending on how one counts) in Fig. 4b. Log_2 of 32 is equal to 5, and log_2 of 31 is slightly less than 5. Four is less than five.

- Examples of "profile information efficiently tailored to identify all bytes of object code executed during the profiled interval, without reference to the binary code of the program" are shown in Figs. 4a and 4c-4f.
- Examples of "non-initiating events" are shown in Figs. 4b and include many of the events denoted by references 534 and 544 of Fig. 5a.
- Examples of "profile circuitry" are shown in Fig. 5a and 5b. A record of a "source and destination of a control flow event in which control flow of the program execution diverges from sequential execution" is shown as arcs 3, 4, 6 and 7 in the top half of Fig. 4a, and profile packets 3, 4, 5 and 6 (each reference 440) of Fig. 4a.

Applicant respectfully submits that no amendment to the drawing is required.

Claim 6 recites that the recorded "profile information [is] efficiently tailored to identify all bytes of object code executed during the profiled interval." In known systems that record profile information sufficient to "identify all bytes of object code executed during the profiled interval," the profile information includes a profile description of every instruction executed. This amount of information is very large, and therefore inefficient. In contrast, Figs. 4a, 4c, 4d, 4e and 4f, and pages 68-72 of the specification show a way of storing profiling information that allows this identification to be accomplished in a much smaller amount of storage. One of ordinary skill, having knowledge of known systems and the contrast with the embodiment shown in the specification, would be reasonably apprised of the scope of the invention.

Claims 3 and 8 recite "circuitry ... configured to record profile information describing at least all events occurring during" a specified time interval that meet specified conditions. This language is not relative – it says that the profile circuitry profiles all events meeting the conditions, and may profile additional events as well.

II. The § 102 Rejection of Claim 1

Independent claims 1 is rejected over the combination of two different embodiments, the prior art discussed in the Background of Chernoff '028 and Chernoff's Description of the Preferred Embodiments. Claim 1 recites as follows:

> 1. A computer, comprising:
> an instruction pipeline configured to execute instructions of the computer;
> profile circuitry configured to detect and record, without compiler assistance for execution profiling, profile information describing a sequence of events occurring in the instruction pipeline, the sequence including a description of every event occurring during a profiled execution interval that matches time-independent selection criteria of events to be profiled, the recording continuing until a predetermined stop condition is reached, the profile circuitry further

> configured to detect the occurrence of a predetermined condition, after a non-profiled interval of execution, and to thereon commence of the profiled execution interval.

As explained below, the claim recites "events occurring in the instruction pipeline," that is, things that happen in hardware. In contrast, Chernoff describes condition codes implemented entirely in software – Chernoff's hardware has no condition codes. Thus, Chernoff's software "condition codes" record things that happen in software, not hardware "pipeline events" as recited in claim 1. Thus, Chernoff cannot meet claim 1.

The Background of Chernoff '028 discusses "condition codes." Chernoff's Background clarifies that the "condition codes" in question are those of the Intel X86. Excerpts from vol. 1 and vol. 2 of the X86 architecture book are attached as Exhibit A. Chernoff states that there are only six condition codes: Carry, Parity, Adjust, Zero, Sign and Overflow. (*see* Chernoff '028 at col. 2 line 66 to col. 3 line 2; Ex. 1 vol. 1 pages 11-12).

The X86 uses these condition codes to implement conditional branches for controlling software program flow. Each arithmetic instruction executed by the processor (ADD, SUB, compare, etc.) sets the condition codes reflecting the result of that instruction – for example, if an ADD causes an overflow, then the Overflow condition code is set, otherwise, it is cleared. If the result of an ADD or SUB is zero, then the Zero condition code is set, otherwise, it is cleared. If the result of an ADD or SUB is negative, then the Sign bit is set, otherwise, it is cleared. (see vol. 1 pages 3-11 to 3-12 and vol. 2 page 3-17). When the X86 reaches a software conditional jump instruction, the conditional jump tests the condition codes to decide whether or not to branch (*see* Ex. 1 vol. 1 pages 6-31 to 6-33 and vol. 2 pages 3-241 to 3-244).

The main description of Chernoff '028 discusses a software emulator for the X86 implemented on the Alpha architecture. (Excerpts from the Alpha Architecture Reference Manual are attached as Exhibit B.) That is, Chernoff discusses a program that runs on Alpha hardware that simulates the X86. As noted by Chernoff '028 at col. 2, lines 59-63, there are no condition codes in the Alpha hardware. For example, the Alpha's ADD instruction does not set any condition codes. *See* Ex. 2 page 4-23.

At col. 3 lines 22-29 and lines 51-61, and at col. 20, line 22 to col. 25, line 10, Chernoff '028 teaches that the X86 condition codes are maintained in software, as information stored in memory. Among the information stored are pointers to "methods," which are no more

than pointers to functions to be executed like any other functions. (See, *e.g.*, col. 36, lines 48-54, discussing C++ – "methods" in C++ are simply functions that are called through pointers.)

In summary, Chernoff's implementation of the X86, including the X86's "condition codes," is entirely in software. Chernoff's condition codes record only events that occur in software, not "events occurring in the instruction pipeline" as recited in claim 1.

Chernoff '028 cannot be said to meet this limitation of claim 1.

Further, because these condition codes are implemented in the X86 hardware, Chernoff's program is "hard coded" around these six condition codes. No user or program has the ability to add a new condition code, such as a condition that might "allow for information profiling to occur." (Office Action, page 5, line 13).

At the very least, no § 102 rejection can be raised, because Chernoff's Background and Description are directed to two entirely different embodiments.

Claim 28 is patentable for similar reasons.

III. Dependent claims

Dependent claims 2-6, 10-11, 14, 17, 19-20, 22, and 24-25, are rejected over the art. However, because claim 1 is patentable over the art, these claims are as well. In addition, the dependent claims recite additional features that further distinguish the art.

IV. Official Notice

Applicant takes note of the Examiner's liberal reliance on Official Notice. MPEP § 2144.03(B) governs the use of Official Notice (emphasis added, citations omitted):

> If [official notice] is taken, the basis for such reasoning must be set forth explicitly. The examiner must provide specific factual findings predicated on sound technical and scientific reasoning to support his or her conclusion of common knowledge. The applicant should be presented with the explicit basis on which the examiner regards the matter as subject to official notice and be allowed to challenge the assertion in the next reply after the Office action in which the common knowledge statement was made.

Applicant traverses reliance on Official Notice because the Office Action does not present "explicit basis on which the examiner regards the matter as subject to official notice." Because no "explicit basis" has been set forth, a mere demand for a reference suffices to traverse Official Notice, pursuant to 37 C.F.R. § 1.104(d)(2). Applicant hereby demands a reference or an affidavit for the following propositions:

- "profile circuitry ...configured to record profile information identifying each distinct physical page of instruction text executed during the profiled execution interval"
- a profile that records an instruction reference "records the event of a page boundary of the address space occurring within a single instruction"
- a profile that records an instruction reference "records the event of a page boundary between two instructions that are sequentially adjacent in the logical address space"
- recorded profile information [that] indicates ranges of instruction binary text executed by the computer during a profiled interval of the execution, the ranges of executed text being recorded as low and high boundaries of the respective ranges

If no reference can be supplied, allowance would be in order.

V. Conclusion

In view of the amendments and remarks, Applicant respectfully submits that the claims are in condition for allowance. Applicant requests that the application be passed to issue in due course. The Examiner is urged to telephone Applicant's undersigned counsel at the number noted below if it will advance the prosecution of this application, or with any suggestion to resolve any condition that would impede allowance.

Enclosed is Petition for Extension of Time for one month. In the event that any further extension of time is required, Applicant petitions for that extension of time required to make this response timely. Kindly charge any additional fee, or credit any surplus, to Deposit Account No. 23-2405, Order No. 114596-08-4015.

Respectfully submitted,

WILLKIE FARR & GALLAGHER, LLP

Dated: October 20, 2003

By: _____
David E. Boundy
Registration No. 36,461

WILLKIE FARR & GALLAGHER, LLP
787 Seventh Ave.
New York, New York 10019
(212) 728-8000
(212) 728-8111 Fax